N-AET
N15-P
NF
[illegible]
277405

# INSTITUTE FOR SPACE STUDIES

P.34

## STELLAR DYNAMICS IN A DISCRETE PHASE SPACE

R. H. Miller

K. H. Prendergast

(NASA-TM-102909) STELLAR DYNAMICS IN A DISCRETE PHASE SPACE (NASA) 34 p

N90-70757

Unclas
00/90 0277405

GODDARD SPACE FLIGHT CENTER
NATIONAL AERONAUTICS AND SPACE ADMINISTRATION

# STELLAR DYNAMICS IN A DISCRETE PHASE SPACE

R. H. Miller

University of Chicago

and

Institute for Space Studies
Goddard Space Flight Center, NASA
New York, New York

and

K. H. Prendergast

Columbia University

## ABSTRACT

A new approach to numerical experiments with self-gravitating stellar dynamical systems is described that permits enough bodies to be included to make the problem interesting. The forces are obtained by solving the Poisson equation at each integration step. The method incorporates the essential physical features of collision-free systems exactly and appears to be adaptable to a wide class of initial value or self-consistency problems. Stellar systems of 120 000 particles have been followed; preliminary results are given.

## I. A GAME

Consider the following game. A point is made to hop about over a two-dimensional lattice according to these rules:

(1) It may reside only at locations whose coordinates $(x, u)$ are integers.

(2) It may hop from one allowed location to another, but must always alternate a hop in which only $x$ changes with one in which only $u$ changes. Let the value of $x$ immediately following the nth step be $x^{(n)}$, the value of $u$ following its next change after $x^{(n)}$ was reached be $u^{(n+1/2)}$,

(3) The value of $x^{(n+1)}$ is given by

$$x^{(n+1)} = x^{(n)} + u^{(n+1/2)} \tag{1}$$

(4) There is a table that gives the rule for changing $u$ according to the present value of $x$. Let the value read out of this table following the nth step be $f^{(n)}$. Then the value of $u^{(n+1/2)}$ is given by

$$u^{(n+1/2)} = u^{(n-1/2)} + f^{(n)} \quad . \tag{2}$$

The pair of moves is a complete step.

As an example of this game (Fig. 1) the particle might start from location "a" with $x^{(0)} = +2$, $u^{(-1/2)} = 0$; the table of f's gives $f^{(0)} = -2$ so $u^{(1/2)} = -2$, as shown at b in Fig. 1. The game, played according to these rules, fills out Table 1. The successive locations occupied by the point (at values of $n$ just following the indicated shifts) are shown in Figure 1. In this case, the point is back at the same location in the lattice at $n = 6$. Thereafter it would endlessly repeat the same set of $(x, u)$ values. The loop in Figure 1 connects those locations occupied just after the shift of the $x$ value ($n$ = integer +; locations a, c, e, g, i, k, m).

Suppose there were many points in such a diagram, each hopping about the lattice according to the stated rules, without regard for the other points. Then each point on a given row of $u$ = const. would be moved the same distance (to the right by the value of $u$) at integer values of $n$ as shown in Figure 2(a). The subsequent moves at half-odd integer values of $n$ would be upward along a column of fixed $x$ by the amount specified by $f$. (Figure 2(b)).

Irrespective of the nature of the table of f-values (even if the table is changed for each n), it is easy to see that (1) The contents of a cell are transferred from one location to another as a unit. Thus, the contents of two cells cannot come to occupy the same location at a later time, nor can the contents of one cell split to occupy two cells. (2) If the process were run backward after n steps (taking care properly to reverse the sequence of operations), all points would return to their original locations.

## II. RELATION TO DYNAMICS

The game just described bears several suggestive resemblances to dynamics. The lattice is like a discrete phase space (distinctions about evaluating x's and u's at different n's aside) for a one-dimensional problem if the identifications $x \longleftrightarrow$ corrdinate, $u \longleftrightarrow$ velocity, $f \longleftrightarrow$ force per unit mass, $n \longleftrightarrow$ time are made. The property that occupants of a lattice location stay together is reminiscent of the Liouville theorem in the phase-space. This system describes the reversible flow of an incompressible fluid in the (discrete) phase space. Since the collision-free Boltzmann equation describes this kind of motion in phase space, the game approximates to solutions of that equation.

Equations (1) and (2) are a finite-difference scheme for integrating $\dot{x} = u$; $\dot{u} = f$, if the value of the time-step be incorporated into the definitions of u and of f to make the entire system dimensionally compatible. The game is like any other technique for numerical integration in that there are higher-order terms ignored in this finite-difference scheme; these will be discussed later.

The game contains the physical features of the phase-space description exactly. It is interesting to have an approximation method that possesses exact integrals of its own. The game carries out an approximate integration without doing arithmetic. The game clearly can be generalized to more dimensions.

The terms ignored in the finite-difference scheme are of the third order in the time-step (this is the object of alternating the velocity and coordinate moves). The assignment of discrete values to the variables corresponds to roundoff while the ignored terms give rise to a truncation error. Details on a scale not substantially larger than a cell size are obviously not to be trusted. Although we have used some more elaborate methods to study this question (such as considering the mapping of a space at one time onto a space a time-step later), the essential features may be easily appreciated by requiring that the truncation error (in the worst case) be nearly the same as the roundoff error; that there be no ambiguity about which location a point should be placed in after a complete step. Again the essential features are included in the one-dimensional formulation. A Taylor series development of $x$ and $u$ as functions of the time leads to:

$$T^3 < \frac{12 \ \varepsilon L}{\left| u \frac{df}{dx} \right| \max} \tag{3}$$

$$T^4 < \left| f \frac{df}{dx} + u \frac{d^2 f}{dx^2} \right|^{-1}_{\max} \cdot 12 \ \varepsilon L \tag{4}$$

where $T$ is the time step, $L$ is the distance between successive lattice points in $x$ (the corresponding interval in $u$ is $L/T$), the derivatives are to be understood as appropriate to the physical system being mimicked, and the quantities marked as absolute values with a subscript "max" are to be worst cases. All quantities (except $\varepsilon$ ) appearing in these equations are dimensioned physical variables. If these relations are satisfied, then the integer location of the point after a complete step will be within a distance $\varepsilon$ L (or $\varepsilon$ L/T) of the location that would be occupied by a particle that moved exactly according to the mechanical equations of motion. Typically, these two relations give similar bounds on the size of the time-step. For reasons of economy and because of the discrete nature of the variables, T should be as large as possible consistent with these relations.

There is no feature of the quantization that affects the stability of the game as a finite-difference approximation to an initial-value problem. This is intuitively evident, particularly on the basis of another representation of the system that will be described in the next paragraph. On more formal grounds, let the time step vary and let L vary with it in such a way that the ration L/T is constant. This has the effect (as T gets smaller)of refining the lattice by putting more and more points in it. The problem is properly posed, and the approximation is consistent and satisfies the sufficient conditions for stability of Lax and Richtmyer (1956).

In this connection, it is instructive to consider another representation of an n-body system that is useful for certain fundamental properties. The $\gamma$ -space (6n-dimensional for a 3-dimensional problem) might also be quantized. An n-body system would occupy some single location in the 6n-dimensional lattice. The 6n-dimensional lattice can be mapped onto a one-dimensional array in which the representation of the system is now a vector with all elements being zero except for one that describes the n-body system. When the system advances through a time-step, the phase point will move to a new location in the $\gamma$ -space, where it will be represented by a new vector all of whose elements, save one, are zero. That one element is again unity. Since the motion in the $\gamma$ -space is completely determined (for a classical system), the transformation of one of these vectors into the other may be represented by a square matrix. Because the motion is determined and time-reversible, the transformation matrix contains just one non-zero element in each row and column. The non-zero elements must be unity. The matrix does not depend upon the system configuration. The result of two time-steps must be capable of representation by a transformation matrix having the same properties. It is clear, since the matrix is finite (for bounded motion), that it is an element of a cyclic group. The order of the group is the number of time steps in a Poincaré recurrence time. This representation makes it intuitively obvious that the game is stable as an initial value problem.

## III. MACHINE CALCULATION

The game was designed to fit easily into the basic operations of a digital computer. Programs have been written to play the game, as a representation of a stellar dynamical system. The first, written for the Maniac III computer at the University of Chicago, used a two-dimensional system (4-dimensional phase space) with 48 x 48 cells in configuration space and 15 x 15 in velocity space and was tried with a single particle in a "harmonic oscillator" force field to explore the programming problems. Later, the full stellar dynamical problem was coded for the IBM 360/75 at the Goddard Institute for Space Studies in New York. The properties of this machine that make it well suited to this problem are 1) large memory (4 million "bytes" of "large core store" = $2^{25}$ bits, plus 1 million "bytes" of fast working storage) and 2) a fast processor. A two-dimensional problem can fit nicely into this store by dividing it into 128 x 128 cells in configuration space, with a 45 x 45 velocity space attached to each configuration space cell. With this selection, the large core is 2025/2048 filled.

The configuration space is made periodic--a particle leaving one boundary is replaced by one entering at the opposite side with the same velocity. A similar recipe in velocity space did not seem feasible; instead, the velocity space was truncated--any particle that acquires $|u|$ or $|v| \geq 23$ is said to have "spilled" and is removed from the phase space array. A running tally of spills is kept.

The system described so far would work for any forces; the forces of interest for stellar dynamics are self-gravitational. Normally, n-body integrations proceed by calculating the resultant force on one particle due to all the others--the pair interaction. This becomes expensive when many particles are involved.

In our calculation, the forces due to self-gravitation are calculated at each step by solving the Poisson equation. Fourier-transform methods are used to reduce the number of calculations from $N^4$ (where N is the number of permitted locations in a periodic length along one configuration direction, N = 128 in this problem) to a multiple of $N^3$. The calculation is reduced by using even and odd parts. Further reductions of the Fourier-transformation process are possible (Cooley and Tukey, 1965); because the force calculation requires a small part of the step time, further economies cannot produce a significant reduction in computation time.

There is a difficulty in the Fourier transform method that caused us some trouble. The periodic configuration space permits Fourier series to be used. Only as many Fourier components need be used as there are lattice points along an axis in the configuration space (128 here). However, truncation of the Fourier series for the density corresponds to a certain density distribution that may be significantly non-zero away from the integer lattice locations. These nearby bumps in the density distribution may produce a greater force than that due to the mass that supposedly generated the density distribution. When we used analytically derived coefficients the forces calculated showed "diffraction patterns" that were manifestly incorrect. The convolution theorem in Fourier transforms may be used to generate the required force on each lattice-point directly by Fourier transforming the force-field about a point mass at the origin. The numerical accuracy is poorer but adequate for the game since the forces must take on integer values, and large values of the force cannot be permitted (Eq's 3 and 4). In the present problem, the forces are restricted to have $|f| < 8$.

It is the fact that the forces need not be determined to high precision, and can thus be obtained from the Poisson equation at a fairly small number of locations that lets the game handle very large numbers of particles. The time required for the force calculation is independent of the number of particles. It is important, however, that the force calculation take proper account of large numbers of particles at substantial distances. This is done by calculating in floating point, and only truncating to integer values at the very last step. The force calculation is the only part of the "integration" that requires arithmetic.

Essentially any desired force-law can be built into this problem by selecting the force field to be Fourier transformed. We use a force between pairs of particles derivable from a potential $-GM/(a^2+x^2+y^2)^{1/2}$. Periodicity is taken into account in the configuration space by including the contribution to the forces from the periodically repeated mass distributions; the gravitational field arising from the mean density is suppressed. The inclusion of $a \neq 0$ is equivalent to considering mass distributions that extend over a few neighboring lattice locations. Among the difficulties that are avoided by using $a \neq 0$, are for example, close encounters and the question of energy conservation. The potential energy of two particles on the same lattice-point is $-GM^2/a$; if $a \to 0$, this becomes infinite, and our system could not even approximate to energy conservation or to the virial theorem without an _ad hoc_ treatment of the potential energy of coincident particles. Since the system cannot relax in the usual sense (through the action of the collision term in the Boltzmann equation), it is not mandatory that we treat close encounters correctly. Furthermore, the Poisson equation--and the Boltzmann equation--both imply some smoothing of the density distribution of point particles. We plan to experiment with different values of a but have not yet done so.

The program, as it now runs, makes about 8 complete steps per hour. About half of the step time is consumed in recording the results. At each step, the entire phase space is copied onto magnetic tape; about five time steps fill a reel of tape. Post-run summaries can be made from the tape copies, but with no need to re-run the problem. The full force calculation, to produce the integer values of the x and y components of the force, each on the 128 x 128 grid, requires about 50 seconds. This is not particularly fast compared to some figures that have been quoted in the literature (Hockney, 1964).

At each integration step, the density, the vector moment of velocity, the velocity ellipse, and the vector force field are plotted for each relevant configuration space cell. There is much more information available--and it is fairly easy to construct subprograms that can form the required quantities from the tape copies of the phase space. Some of the first quantities that we expect to form include the phase density, $f(\mathcal{E}, J, \varphi)$, as well as a decomposition of the motions into wave-like and convective parts.

Rather than removing the "spills" from the problem as we have, it would be possible to keep track of them and to integrate their orbits more carefully. The small number of "spills" obtained in the examples run so far indicates that our present treatment presents no difficulty for reasonably scaled problems.

Other boundary conditions could easily be substituted for those we use. The system might have reflecting barriers in velocity space, or in configuration space, or the periodic condition in configuration space might be dropped, treating particles that reach the configuration boundary as "escapes", just as those that have reached the velocity boundary are spills.

Relaxation could be incorporated into the problem by admitting scatterings with a random property. This might be done in any of several ways, but one simple way would be to rearrange some of the velocities of particles in a given configuration space cell, in a random way. This can be done in a manner that will conserve the physical integrals (energy, momentum) to within the accuracy allowed by the admissible velocity values.

The game might be played using different storage representations. The one chosen seems well suited to a Fermi gas, where there is an *a priori* exclusion to prevent more than one particle per phase space cell, but stellar problems are realistic only if the density is so low that the distortion caused by limiting to one particle per phase cell is negligible. With the present 33 million phase space cells, we feel comfortable inserting $10^5 - 10^6$ particles. We considered partitioning the memory in such a way as to permit several bits to represent each cell--an 8-bit "byte" being a reasonable unit. Then the extra bits might represent some attribute of the phase space cell--different masses, if only one particle per cell is permitted, or different numbers of particles. We felt that finer-grained treatment of larger numbers of equal mass particles would be more informative, and designed the program accordingly. Another memory allocation that has many desirable properties for this problem is to use list storage; a scheme that uses all available storage fairly well, but which encounters a certain storage and processor overhead.

## IV. PRELIMINARY RESULTS

So far, we have tried several problems, all of which are indicated in Table 2. Because the program is new, new results are coming rapidly, and Table 2 only indicates the situation at a certain date. Each problem run suggests several new ones, and each way of summarizing the results raises questions that require new summaries. We hope to prepare more detailed reports of the results and of the very interesting physical implications, but for the present must restrict ourselves to a qualitative description of the results, particularly as they reflect upon the validity of the approximation.

Scaling the problem is the most delicate part of starting a run with a new set of initial conditions; a set of initial conditions for $10^5$ - $10^6$ particles requires some care. Internal motions and a velocity dispersion must be supplied for each configuration space point. Normally almost any initial condition is pathological, and changes quickly as the integration proceeds; there seems to be little point either in selecting bizarre initial conditions or in avoiding rapid changes through the use of stationary self-consistent models. The quantities that may be specified as the initial conditions are (1) the value of GM, (2) the number of particles in a configuration space cell, and (3) the distribution of those particles in velocity space to represent the desired mean velocity and velocity dispersions. Effectively, the choices of GM and of the number of particles in a configuration space cell ($\sigma$) sets the time-scale; the value of G$\rho$ that would correspond to a given set of conditions is essentially GM $\sigma/a$, where a is the quantity that appears in the force calculation. We find it most convenient to set the adjustable parameters empirically. The process of scaling the problem starts with a projection of the desired model onto configuration space followed by a calculation of the resulting forces at every lattice point. The constant GM is chosen to yield a greatest value for a force component of about 3, and internal motions are chosen according to some simple pattern approximately to balance the gravitational forees. A machine program is then written to load the phase space array, and checked separately before being used with the main integration routine.

The first problem run was the "quiet circle." It has a very high degree of symmetry--exact fourfold symmetry in the initial condition. The retention of the exact fourfold symmetry is a valuable check on the routine. At 25 steps, the fourfold symmetry is still exact. Somewhere, by 60 steps, different rounding to integer values in the force calculation has lost some of the symmetry--it now shows a twofold symmetry. This means that inaccuracies of about $10^{-6}$ - $10^{-8}$ are appearing in our numerical processes, and that the calculation preserves the exact integrals of the approximation to a satisfactory degree. (The spilling of a number of particles not divisible by 4 shows that the symmetry was lost somewhere along the line). Another check on the integration process is by comparing the results (as number-densities projected onto the configuration space) between the "quiet" and "noisy" cases, the "noise" being a perturbation in the velocity space. These problems, in both cases where comparisons could be made, remained reassuringly similar out to about 18-20 integration steps. In Table 2, the noise is indicated by the + R term in the <velocity> entry; R is + 1, 0, or -1 with equal probability, and is produced by a pseudo-random number generator.

These systems feel the presence of identical systems repeated over the periodic lattice; they tend to stretch out preferentially toward their nearest neighbors. We could remove the explicit neighbor-interaction (with the Fourier series force calculation, there will always be effects due to the periodicity present), which should reduce this effect. This is one of many things we hope to try sometime. The rectangular systems show the neighbor interaction especially strongly.

The circle problems show an interesting initial collapse, followed by the development of a hole in the center. Later, the system collapses again, only to develop the hole anew--although with a much more complex structure. The central condensation is greatest at about 6 steps and again at about 18 steps. The hole is most pronounced at about 12 steps and opens again at about 24 steps. All the time, clear differential rotations persist, and the background around the central mass fills up with evaporated particles. "Spiral" patterns develop, but are short-lived. After 40-50 steps, the systems settle down to something that looks like a steady state, with very little further development. About half of the particles are in the central mass, and the other half filling the general background. Densities (number of particles in a cell of the configuration space) run to about 170 near the center, and drop off to about 4-5 in the outer region. The central peak is about 10-15 cells wide (diameter at half maximum). In the "noisy circle", this final state appears to be a rotating ellipse, while the symmetry of the "quiet circle" forces a circular shape. The fourfold symmetry forced by the neighbors is lost in the noisy circle problem after about 20-30 steps.

The rectangles tend to build bridges to the next periodic cell. The fast rectangle rotated fast enough that it tore itself into two pieces which fly about and collide with one another several times in the 51 steps run. The pieces interpenetrate.

The "Jeans problem" was run to see whether we could mimic the "Jeans instability" with this system. The force constant, GM, was set very large to assure that there would be nonzero forces somewhere in the system. The initial state was loaded by a pseudo-random number generator. Because the force was so large, spills followed as soon as a pattern developed. In about 3 steps the pattern is very well developed, at 7 steps the valleys are beginning to fill in, but by then 1/10 of the initial particles have spilled. The wavelength of the pattern that built up was about one full wave per priodic cell; the Jeans length for this problem is about 8-16 small cells (1/16 to 1/8 of a periodic cell). We cannot make quantitative comparisons with the "Jeans instability" at this time because our time step was too large. We hope to carry out further experiments with this problem.

The approximation conserves its integrals very well, and appears to conserve the usual integrals of the mechanical system being mimicked better than we had expected. Because the game matches the physics of collision-free motion exactly, the development of the systems is probably a reasonably faithful representation of a physical system. We are very pleased with the evident good behavior of the game, and anticipate a fruitful application to stellar dynamical problems.

Although the game was devised as a method of attacking stellar dynamical problems, it is perhaps even better suited to other kinds of problems. The representation used really describes a Fermi gas, and it is natural to think of applying this technique to crystal physics. Applications to hydrodynamics are evident. Another possible application is to turbulence theory, representing a wave-number space. In all of these problems, meaningful results require the introduction of interactions between "particles", fixed force fields, and other features that are easy to do, but which we have not yet done.

The work reported here was started at the Institute for Computer Research at the University of Chicago, where it was supported by the U. S. Atomic Energy Commission under contractAT(11-1)-614. The work was completed while one of us (R. H. M. ) held a National Research Council Senior Postdoctoral Resident Research Associateship supported by the National Aeronautics and Space Administration. The hospitality of

Dr. Robert Jastrow and of the Goddard Institute for Space Studies is gratefully acknowledged. Invaluable programming assistance was provided by Naomi Alton in Chicago and by Marian Sackrowitz in New York.

## APPENDIX I

### SCALING TO PHYSICAL UNITS

According to the scaling chosen, a calculated system can represent any of a number of physical systems. There are three physical units to be determined, but the requirement that the gravitational constant take on the correct physical value limits the free choice to two. In principle, any two physical quantities (of different dimensions) might be chosen, but it appears that selecting the density and velocity units may be most convenient.

The scaling of physical units is determined in the initial conditions. These usually consist of a constant number of particles per configuration space cell ($\sigma$) inside some boundary. There is a numerical coefficient $\gamma$ in the force calculation ( $\gamma$ is effectively GM made dimensionless); let K be the largest numerical value that results from a force calculation with both $\gamma$ and $\sigma$ set to unity. Then $\gamma$ is chosen so that $\gamma\sigma K \approx F_{max}$, where $F_{max}$ is the largest value of the force to be allowed in the initial condition (about 3).

The physical time step is determined by the density in the corresponding physical system: it is $1.5 \times 10^7 \, (\gamma\sigma / \rho)^{1/2}$ in years, with the density expressed in solar masses per cubic parsec. If the unit of velocity be $v$ (km/sec), then the units of length (the distance between successive lattice points in configuration space) are $L = 15(\gamma\sigma)^{1/2} v/\rho^{1/2}$ parsecs, and $m = (3400/\sigma)(\gamma\sigma)^{3/2} v^3/\rho^{1/2}$ solar masses. The mass $m$ is the mass of one of the "particles" that is moved about in the game. It may be regarded as an aggregate of stars, rather than as a single star, for some purposes.

As an example, the rectangle and circle problems have $\gamma\sigma \approx 1$; they may be scaled to the solar neighborhood taking $\rho \approx 0.15\ M_{\odot}/pc^3$, so the time unit is $3.9 \times 10^7$ years. Sixty steps of evolution then represent about $2.5 \times 10^9$ years. Making the largest circular velocity (10 units) match the rotational velocity in the solar neighborhood (250 km/sec) sets the length $L$ at about 1000 pc. The 37 masses initially in each configuration space cell are each of about $4 \times 10^6\ M_{\odot}$, and the total of 120 000 particles represent $5 \times 10^{11}\ M_{\odot}$. The periodic length is about 125 kpc, and the rms peculiar velocities (2.4 units) are about 60 km/sec.

# APPENDIX II

## THE PROGRAMS

For the most part, program details are available from internal documentation. However, the general structure of the program, and of its principal parts, may not be evident from the listings. This appendix is intended to give an overall picture of the program organization to facilitate reading of the program listings.

The program is basically a loop consisting of three main parts (Figure A1), those in which (1) the particles are shifted along a coordinate direction at fixed velocity (Figure 2a; labelled "advance coordinates" in Figure A1), (2) the forces are calculated for each configuration space lattice point, and (3) the particles are shifted along a velocity direction at fixed coordinate (Figure 2b; labelled "advance velocities" in Figure A1). The stage at which the phase array is copied onto tape is the stage at which all summaries are made. It is important to note that this is done after the coordinates are advanced, as the interpretation of velocity plots requires knowing that the plot labelled "n" refers to velocities at "n-1/2". The step number is augmented after the "copy on tape" stage. The entry point is so located that the process can be re-started from a tape copy of the phase space.

Two arrays dominate the discussion. PHASE is the array of bits that represents the phase space. It can be declared in FORTRAN as PHASE (4,128,2025). A bit belonging to the coordinates x, y, u, v will be located at word (word=32 bits) address

BASE + x/32 + 4*y + (4*128)*u + (4*128*45)*v

(BASE includes the obvious subtraction to avoid the FORTRAN start from 1 indexing). The location corresponding to given x is the bit consecutively numbered from the x=1 position. The divide is in the sense of an integer divide. CONFIG is a 128*128 array of integers that contains the projection of PHASE onto configuration space. We use the convention throughout with 2-index arrays referring to coordinates (CONFIG, FX, FY) that the first index represents x, the second represents y. FX and FY are integer arrays that contain the values of the force components to be used at each (x, y) point.

The program breaks into two parts: that necessary to establish the initial state of the running routines, which is run only once to get the process started, and the main integration loop. A flow chart of the startup is just a straight line proceeding through the operations. A list of them, performed in sequence, follows:

| | |
|---|---|
| SPILL = 0 | (SPILL is the tally of spills) |
| Clear PHASE and CONFIG arrays | |
| CALL CONSTR (MCOS, MSIN, MFOR, FX, CONST) | (CONSTR is a routine to load the arrays MCOS (co~~s~~sines for Fourier Transform), MSIN (sines for Fourier transform), and MFOR (convolution coefficients) for the force routine, with the value of GM specified by CONST. FX = one of the force arrays, us as a temporary storage in the process). |
| CALL SETINT (PHASE, N, M) | (SETINT initializes SETONE routine; SETON is called by LOAD to place bits into the array PHASE at apecified x, y, u, v values) |
| CALL LOAD(CONFIG) | (LOAD generates the pattern of bits to be set into PHASE; LOAD calls SETONE to do th actual loading) |

| | |
|---|---|
| OUTPUT | At this stage, some features of the initial state are printed and the initial phase space array is copied onto magnetic tape, through some special routines as well as the output routines used at each stage of the loop. There is no fixed output routine.. |
| Enter loop. | The loop is shown in Figure Al. |

The routines appearing in the loop are indicated in Figure Al. As just noted, there is no fixed output routine. The principal output functions at run time are to copy the PHASE array onto magnetic tape and to prepare those plotter tapes that are generated at run time. Certain printed summaries are also prepared (number of spills, step number) at this stage. The calls and functions of the other routines are as follows. SHIFTI is the routine that does circular bit shifts in x, y at fixed u, v. The shift along x at fixed y, u, v is a circular shift of 128-bit objects. The shift along y at fixed x, u, v is a relocation of 128-bit "words" in memory, exactly like the circular shift. The projection of PHASE onto CONFIG is done by SHIFTI. (Call is CALL SHIFTI). SCRBLE is the routine that moves bits in u, v at fixed x, y according to FX and FY. The u, v moves are made simultaneously. Moves are made in a direction opposite to the forces so that bits are always moved into locations that have been emptied earlier in the current step. The bit location to be moved is examined; if it is zero, the move is not carried through. A gain in running speed is made by copying the parts of PHASE that SCRBLE is working on into fast memory. (Call is CALL SCRBLE (PHASE, FX, FY, N, M, SPILL)).

FORCE (Call is CALL FORCE (CONFIG, FX, FY, MCOS, MSIN, MFOR)). The force calculation is rather complicated. It proceeds by Fourier transformations. The Fourier transform is actually a complex transformation written in terms of sines and cosines; only those parts that contribute to the final result are retained. The calculation is reduced by decomposition into four parity states (the numerical coefficients that belong with this process are absorbed into the coefficient that is used in generating the matrix MFOR). These are each Fourier transformed by matrix-multiplication through a BAL subroutine MTRXMY (A, B, C) that forms the 65 x 65 product of A*B and placed the result in C. The BAL routine is about four times faster than a corresponding (CAI/BPS) FORTRAN routine. Since both the x and y force component calculations proceed from the same Fourier tranformed density, the transformed density is saved. With the parity decompositions, the borderir rows and columns of the Fourier-transformed (or re-transformed) arrays must be doubled. This is done in the matrix MFOR for the intermediate (Fourier transform of density) stage, but must be done explicitly after the re-transformation to configuration space and before the re-composition from parity states. Bordering rows or columns that contain all zeroes because of their parity states are not doubled. On the recombination, the force values are rounded to integer values, and restricted to maximum values (magnitudes) of 7. Should it become necessary to flag cases in which the force exceeds this limit, this is the part of the routine in which that should be done. The sines an cosines were stored in matrix form--although there is a great deal of redundan because they are most rapidly available that way. The force calculation requir about 50-55 seconds.

Among the auxiliary programs, two are useful before the loading to help set the scaling (the value of CONST in the starting routine CONSTR).

One of these does the force calculation for a given array CONFIG, and prints the values of the forces at each of the (16384!) lattice points as a floating point number, with GM=1. The second takes the subroutine LOAD(CONFIG) exactly in the form required by the main program, and forms tallies of (1) the total number of particles loaded, (2) the sum of squares of the velocities, (3) the angular momentum about the lattice-point (64, 64) as origin, (4) the maximum and minimum values of x, y, u, v (in the FORTRAN index convention), and (5) the summed velocities (u-23) and (v-23) for all particles. LOAD and SETONE take the integer values of x, y, u, v to conform to FORTRAN indexing conventions of allowing only natural numbers; hence the range on x and y is 1 to 128, that on u and v is 1 to 45, with u or v = 23 corresponding to zero velocity.

Table 1. MOVES IN THE GAME

| n | x | u | f | label |
|---|---|---|---|---|
| -1/2 | | 0 | | |
| | | | | a |
| 0 | +2 | | -2 | |
| | | | | b |
| 1/2 | | -2 | | |
| | | | | c |
| 1 | 0 | | 0 | |
| | | | | d |
| 3/2 | | -2 | | |
| | | | | e |
| 2 | -2 | | +2 | |
| | | | | f |
| 5/2 | | 0 | | |
| | | | | g |
| 3 | -2 | | +2 | |
| | | | | h |
| 7/2 | | +2 | | |
| | | | | i |
| 4 | 0 | | 0 | |
| | | | | j |
| 9/2 | | +2 | | |
| | | | | k |
| 5 | +2 | | -2 | |
| | | | | l |
| 11/2 | | 0 | | |
| | | | | m |
| 6 | +2 | | -2 | |

TABLE 2. EXAMPLES RUN

| NAME | "JEANS PROBLEM" | QUIET CIRCLE | NOISY CIRCLE | FAST RECTANGLE | SLOW RECTANGLE QUIET | SLOW RECTANGLE NOISY |
|---|---|---|---|---|---|---|
| Value of GM | 1 | 1/40 | 1/40 | 1/40 | 1/40 | 1/40 |
| Value of a | 3 | 3 | 3 | 3 | 3 | 3 |
| Max $\|F\|$ Initial Condit. | 2 | 3 | 3 | 3 | 3 | 3 |
| Number of "Particles" | 100 000 | 119 621 | 119 621 | 122 877 | 122 877 | 122 877 |
| Number of Steps Run | 7 | 61 | 59 | 51 | 22 | 18 |
| Spills At End | 10110 | 7 | 15 | 0 | 88 | 3 |
| Density in Initial Condit. | Pseudo-random; Poisson distr. in 16384 locations $\approx 6.15$/config. cell | 37 per config. cell: all cells $x^2+y^2<1026$ | 37 per config. cell: all cells inside $x^2+y^2<1026$ | 37 per config. cell: all cells $-40\leq x\leq +40$ $-20\leq y\leq 20$ | 37 per config. cell: all cells $-40\leq x\leq 40$ $-20\leq y\leq +20$ | 37 per config. cell: all cells $-40\leq x\leq +40$ $-20\leq y\leq +20$ |
| <Velocity> = internal motions | NONE | circular: $U_0=-\frac{y}{r}(r^2/100)$ $V_0=+\frac{x}{r}(r^2/100)$ | circular noise $U_o=-\frac{y}{r}\frac{r^2}{100}+R$ $V_o=+\frac{x}{r}\frac{r^2}{100}+R$ | rotating and shearing $U_0 = -0.3y$ $V_0=+0.4x$ | rotating and shearing $U_0=-y/5$ $V_0=+x/7$ | rotating and shearing+noise $U_o=-y/5+R$ $V_o=+x/7+R$ |
| Velocity ellipse: 1 particle per phase space cell | Maxwellian $\langle u^2\rangle+\langle v^2\rangle=100$ $\|u\|, \|v\|\leq 22$ | circle $\langle \Delta u^2\rangle+\langle \Delta v^2\rangle$ $\leq 11$ | circle $\langle \Delta u^2\rangle+\langle \Delta v^2\rangle$ $\leq 11$ | circle $\langle \Delta u^2\rangle+\langle \Delta v^2\rangle$ $\leq 11$ | circle $\langle \Delta u^2\rangle+\langle \Delta v^2\rangle$ $\leq 11$ | circle $\langle \Delta u^2\rangle+\langle \Delta v^2\rangle$ $\leq 11$ |
| Plots available | Density | Density Force, velocity velocity ellipse | Density Force, velocity velocity ellipse | Density | Density | Density |

## REFERENCES

Cooley, J W., and Tukey, J. W., 1965 Math. Comp. **19**, 297-301.

Hockney, R. W., 1965, Journ. Assoc. Comp. Mach. **12**, 95-113.

Lax, P. D., and Richtmyer, R. D., 1956, Comm. Pure and Appl. Math. **9**, 267-293.

## FIGURE CAPTIONS

Figure 1. Moves in an example of the game.

Figure 2. Systematic moves of many particles in the game. (a) the moves in which $x$ is changed according to the present value of $u$. (b) the moves in which $u$ is changed by the amount of $f(x)$.

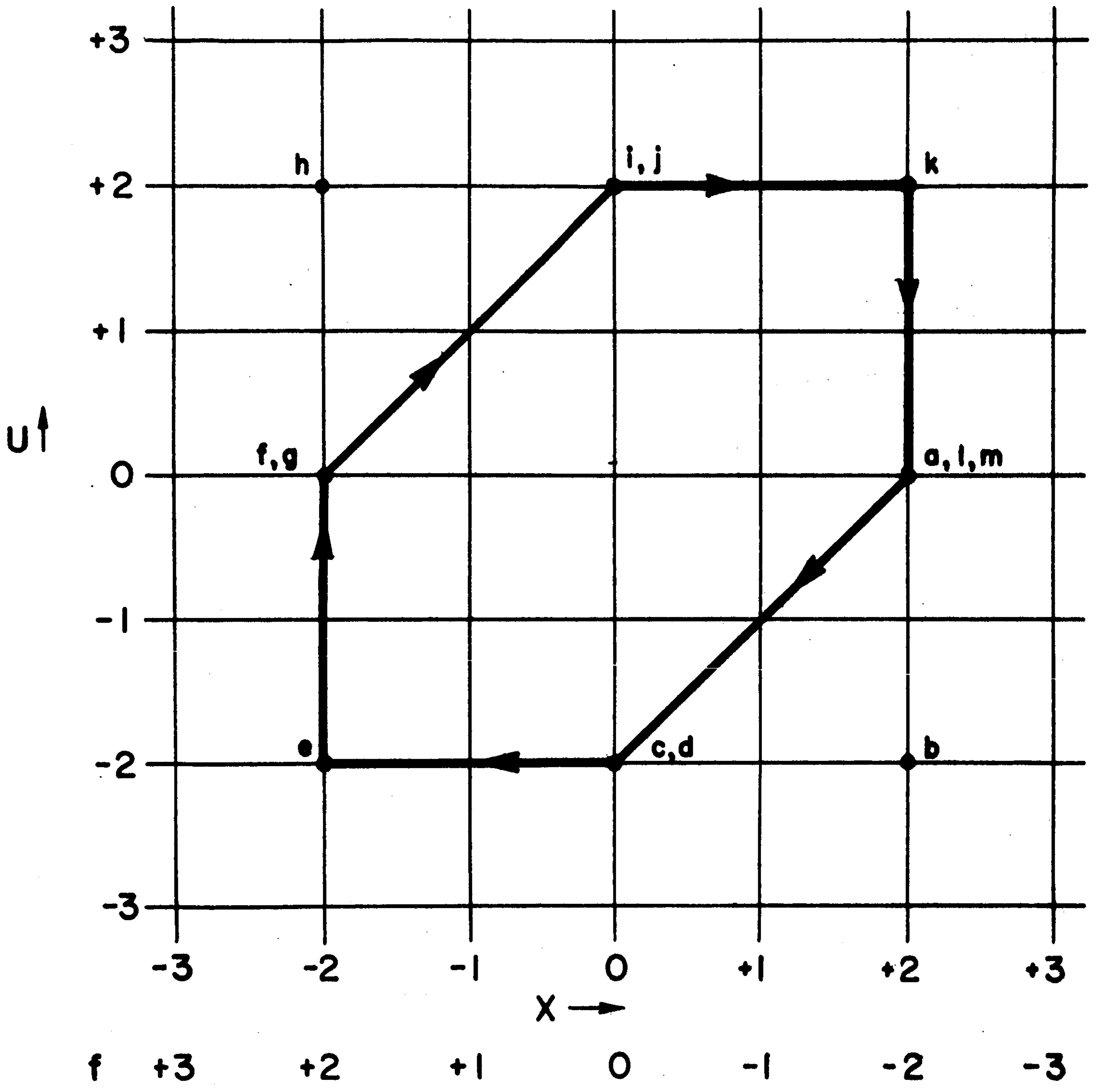
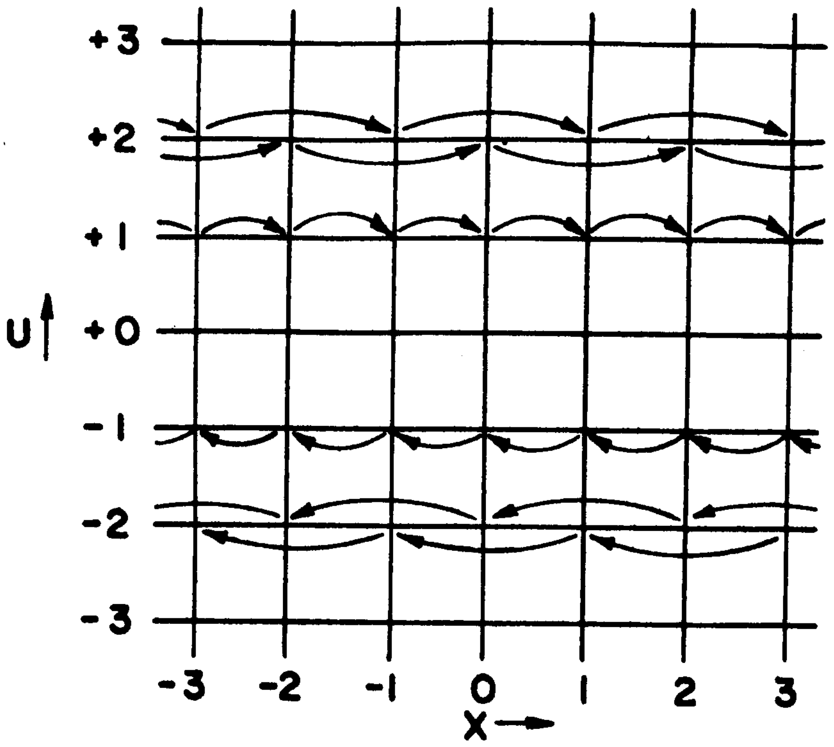
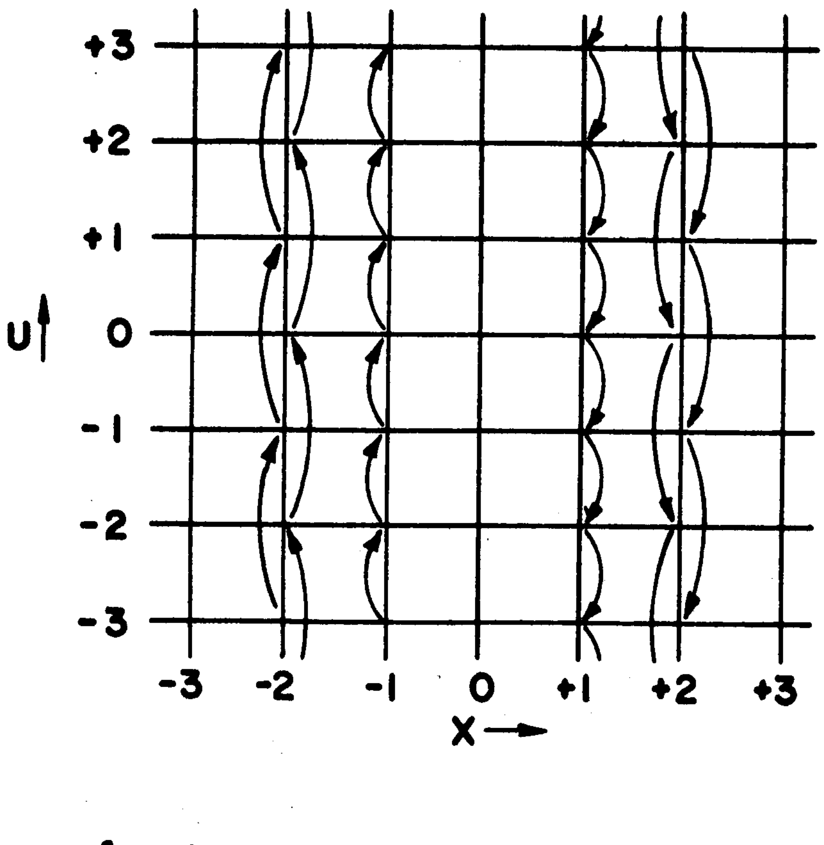


Fig. 1



a.

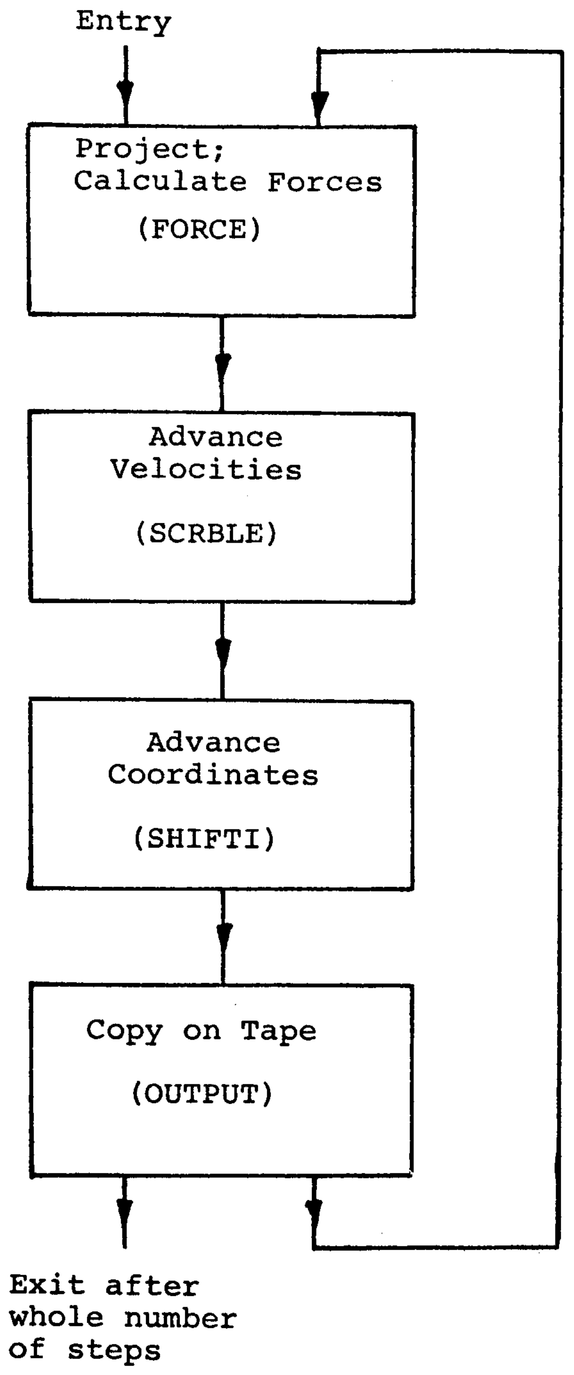


b.

f +3 +2 +1 0 -1 -2 -3

Fig. 2

Figure A1. Flow chart of computer programs for the game.



Entry
Project;
Calculate Forces
(FORCE)
Advance
Velocities
(SCRBLE)
Advance
Coordinates
(SHIFTI)
Copy on Tape
(OUTPUT)
Exit after
whole number
of steps

Figure A1